Lightning Fast Legal Overview

Project Report
Digital Data Science
SEGES

November 1, 2018

SEGES
Landbrug & Fødevarer
F.m.b.A
Agro Food Park 15
DK - 8200 Aarhus N
Phone: +45 8740 5000
info@seges.dk
seges.dk

Summary

Danish Summary

I dette projekt præsenterer vi funktionaliteten af en nærmeste nabo (NN) dokumentsøgning af skatteafgørelser hentet fra world wide web. Vi beskriver intuitionen bag de metoder, der anvendes til at udvikle sådan et kunstig intelligens (AI) system, samt med et designforslag til et endelig system, der er værdifuldt for slutbrugeren. For at evaluere kvaliteten af vores søg-funktionalitet udvikler vi en prototype og annoterer to datasæt med afgørelser. Vores resultater viser, at vores dokumentsøgning kan bruges som et supplement til den traditionelle søgeordssøgning, da vores metode ikke er bedre end den traditionelle metode, dog har vores metode fordelen at være uafhængig af velvalgte søgeord fra brugeren.

English Summary

In this project we present the functionality of a nearest neighbor document search of tax verdicts scraped from the world wide web. We describe the intuition behind the methods utilized for developing such an artificial intelligence system, along with a proposal for the design of a final system valuable to the end user. To evaluate the performance of such search functionality we develop a prototype and annotate two sets of verdicts. Our results show that our nearest neighbor document search is a feasible supplement to the traditional keyword search, as it do not outperform the traditional keyword search. However, our nearest neighbor document search has the benefit of being independent of well chosen keywords by the user.

Contents

1 Introduction

A lot of legal verdicts and laws that impact the daily operation of farms are constantly issued. It is difficult, both for the farmer and his agricultural advisor, to plan the optimal response to a given legal issue. In particular, it is difficult to assess if a given response is in line with related verdicts, given the sometimes quite tangential nature of the relationship between legal cases. New data analysis tools, using artificial intelligence (AI), are able to process a large set of legal documents much faster than any human. Such data analysis tools have the potential to find patterns and systematize the legal documents in such a way that the human can clearly get an overview of what the current practiced law is. The farmer or his advisor may, with such a system, be able to quickly decide on the optimal way to handle a legal issue, thereby saving time and money. As part of this project, legal decisions and verdicts for a given domain are given as input to our AI system with the purpose of assessing whether such a tool can find patterns for the current legal practice to ensure a good foundation for choosing the best legal response.

The current process for a legal advisor, when given a new case, is to first understand the case and find similar verdicts that can strengthen the case in favor of the farmer. However, this process of finding similar verdicts can be difficult as the corpus of verdicts is large and growing.

The focus of this project regards the legal field of tax exemption when selling off a house, a farmhouse, or a summer house. In particular, we will develop an AI system that can quickly and accurately present the previous verdicts relevant to a current legal case in question. This system can be seen as an optimized tax verdict search engine, where the input, given by the user, is a description of the case, and the output is a list of previous verdicts sorted by relevance to the case. To develop such system we consider the following problem statements:

- How can the tax verdicts, publicly available on the world wide web, be given as input to an AI system?
- How can an AI system predict the relevance of previous verdicts to a new legal case?
- How accurate are such predictions?
- Can such an AI system provide better assistance than the current search results provided by the individual verdict websites?

2 Methods

In this chapter we describe the AI method (Section 2.1), utilized by the proposed system (Section 2.1). Our prototype system is develop based on the collected tax verdicts publicly available on the world wide web (Section 2.3) and its performance is evaluated (Section 2.4) using our own annotated data sets (Section 2.5).

2.1 Text vectorization

Human language has no intrinsic meaning from a computer's perspective, thus a word, a sentence, or indeed a whole document is as good as any other, as it is just a collection of characters that make it up. To analyze language and meaning, i.e. semantics, one must first find a way to represent its constituents as mathematical objects, which is the main objective in the field of natural language processing (NLP).

To develop our AI system we use text vectorization methods, which is a sub-field of NLP. Text vectorization methods transform the contents of a document into a vector, i.e. an ordered list of numbers. This text vector inherits the mathematical properties of vector spaces, and key to this approach is the ability to measure distance between such document representations. Just as humans are able to intuitively understand whether the contents of two documents are similar, relatively small distances in the representation vector space indicate similarity of the documents Thus, text vectorization is useful as our AI system relies on being able to compare the content of documents to determine which documents are similar.

Multiple different methods exist for producing text vectors , and for this project we utilize two of the most common, namely term frequency–inverse document frequency (TFIDF) and paragraph vector (PV), with the purpose of evaluating which is best for capturing the between-document similarity. The following Section 2.1.1 and 2.1.2 respectively, will describe the two methods, with a focus on their advantages and shortcomings. Additionally, we also utilize and evaluate two different distance metrics, namely euclidean and cosine distance. These metrics are descried in Section 2.1.3.

2.1.1 Term frequency–inverse document frequency

Term frequency–inverse document frequency (TFIDF) is an algorithm which takes a corpus of documents as input and outputs a text vector per document, where the size of each vector is equivalent to the number of unique words across all documents. TFIDF is an extension of basic term frequency vectorization, where the occurrence of each word in a document is counted and the resulting text vector then contains counts of each word. The extended part of TFIDF is the inverse document frequency, where each word of a document is weighted by how rarely the word appears across all documents in the corpus. The resulting text vectors place less weight on common words (e.g. "a" and "the") and consequently lets rarer words express what the document describes[1].

A disadvantage of TFIDF is that the order of words are lost. Thus, the method belongs to the Bag of Words style of representational methods[2]. This becomes especially pronounced when using the method on corpora with very large documents, as the relation between the first and the last word is weighted the same as two words right next to each other. Additionally, using the full TFIDF vectors for analysis can become extremely

[1] http://scikit-learn.org/stable/modules/feature_extraction.html#tfidf-term-weighting
[2] http://scikit-learn.org/stable/modules/feature_extraction.html#the-bag-of-words-representation

computationally demanding, as the number of parameters equals the number of unique words in the corpus, easily achieving vector sizes in the hundreds of thousands. On the other hand, computing the TFIDF vectors themselves is very efficient as they only require a singe loop over all of the tokenized documents.

In this project, we tokenized and vectorized the corpus documents using the TFIDF implementation in the Python library *scikit-learn*[1] version 0.19.1, with all default settings. Executing this TFIDF implementation resulted in text vectors of size 339.188.

2.1.2 Paragraph vector

Paragraph vector (PV) is a newly (i.e. 2014) developed text vectorization method which utilizes a feedforward neural network (FNN) for training a model that predicts the identity of a document given a window of words from that document. The training of such a model results in a weight matrix optimized for discriminating between documents. The surprising result of this model is that such weights can be used as text vectors because they capture the semantic relations between documents[2], which is not possible for methods like TFIDF.

However, a disadvantage of PV is that it is more computationally expensive than TFIDF. Additionally, it can be difficult to know when the training procedure has converged, since the whole process is unsupervised and questions of optimality are hard to answer. Some of the hyperparameters to optimize are for instance, the number of epochs (i.e. number of training iterations over the corpus) and the size of the hidden layer in the FNN (which determines in size of the text vectors, as they become the same size).

In this project, we used an off-the-shelf PV implementation from the Python package *gensim* version 3.4.0. This implementation is called *doc2vec* and we used its default settings, except we opted for a text vector length of 200 and ran the training algorithm for 100 epochs. Additionally, the execution utilized 20 workers and a random state set to 0. The tokenized corpus given as input to doc2vec was tokenized using the `word_tokenize` function from the Python package *nltk* version 3.2.5.

2.1.3 Vector space distance metrics

To measure the distances between text vectors we use two difference distance metrics, namely euclidean and cosine distance. Conceptually, the difference the of two distance metrics is that euclidean distance captures the familiar notion of a straight line between the vectors, whereas cosine distance captures the angle between the two vectors. This conceptual difference is also illustrated in Figure 2.1 between the 2-dimensional vectors A and B.

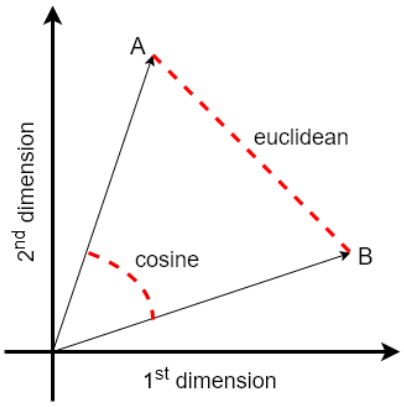


Figure 2.1: Visualization of difference of the euclidean and cosine distance metrics.

In the context of high-dimensional text vectors, different distance metrics often do not have a linear mapping between them, i.e. a large euclidean distance between two vectors does not necessarily imply a large cosine distance between the same two vectors. For example, using euclidean distance on numerical bag-of-word text vectors (e.g. TFIDF) is often said to be misleading, as distances between long documents and short documents will be skewed; long documents often contain a larger vocabulary of words than short documents. However, the same can not be said for PVs as the distance between long and short documents depends on the trained models ability to embed the documents with respect to the content they describe and not the vocabulary they use.

In this project, we used the euclidean[3] and cosine distance[4] functions implementations in the *scikit-learn* version 0.19.1, with all default settings. Note that the cosine distance is obtained by subtracting the result of the function `cosine_similarity` from one, resulting in values ranging from 0 to 2.

2.2 The proposed AI system

Having described the AI methods utilized by our proposed system, we can describe the full design of the prototype system developed in this project, see Section 2.2.1. This prototype system is not usable for the end user, i.e. the legal advisors, thus in Section 2.2.2 we describe our design ideas for a production system with the end user in mind.

2.2.1 Prototype system

Our prototype system consists of two individual parts. The first part creates the text vectors by the workflow illustrated in Figure 2.2. In this workflow we start off by scraping the tax verdicts from the websites. All of these verdicts in plain text are then given as the corpus to the two text vectorization methods. The outputs from the text vectorization methods are then created as text vectors in each vector space, respectively.

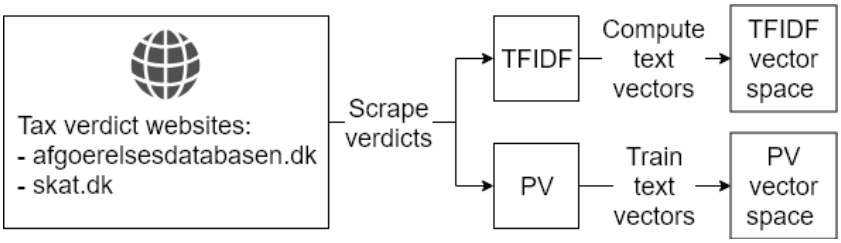


Figure 2.2: First part of our prototype system.

The second part of our prototype system facilitated the text vector powered verdict search engine through the workflow illustrated in Figure 2.3. In this workflow, we start by selecting a verdict from which we wish to find related verdicts. Note that the selected verdict has to been present in the scraped data set. This selected verdict is given as input to the search engine, which finds the precomputed text vector for the input verdict and computes the neighboring verdicts in each of the two vector spaces, based on the one of the two distance metrics. The neighboring verdicts are then returned as the search results, thus the verdict search is a nearest neighbor (NN) search of verdicts in the vector space.

One of the purposes of this prototype was to determine which text vectorization method and distance metric best captured the relevance between verdicts. Therefore, the prototype returns four sets of neighboring verdicts, i.e. one per combination of text vectorization method and distance metric. Based on these result sets and the annotation data set,

[3] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.euclidean_distances.html
[4] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html

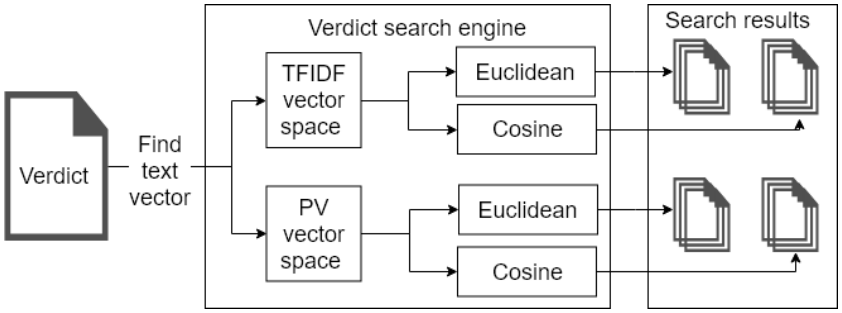


Figure 2.3: Second part of our prototype system.

described in Section 2.5.1, we can evaluate which combination best captures the relevance between verdicts.

2.2.2 Production system

We propose a production system usable for the end user, based on our prototype system and our evaluation of the best text vectorization method and distance metric. This proposed system consists of three individual parts inspired by our prototype. The first part is mainly a data engineering task of scraping all legal documents from the world wide web on a periodically basis, and merging all of the data sets into a single database. This workflow is illustrated in Figure 2.4.

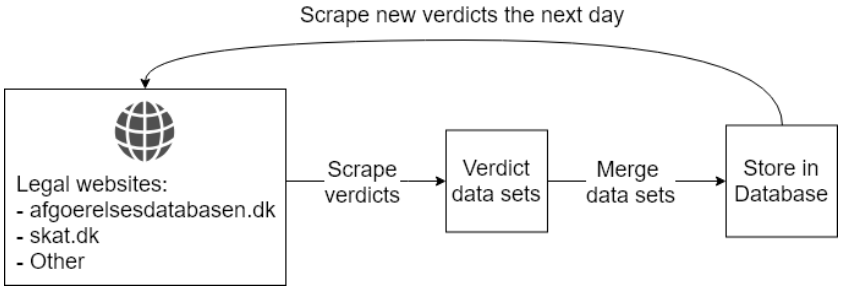


Figure 2.4: First part of our proposed production system.

The second part concerns the the continual computation of the text vectors, due to the periodically update with new documents in the document database. As illustrated in Figure 2.5, our text vectorization method fetches the documents from the database. We suggest that the computation of text vectors consists of a daily inference of the new documents and a weekly recomputation of all vectors, including a performance evaluation to ensure high quality vectors.

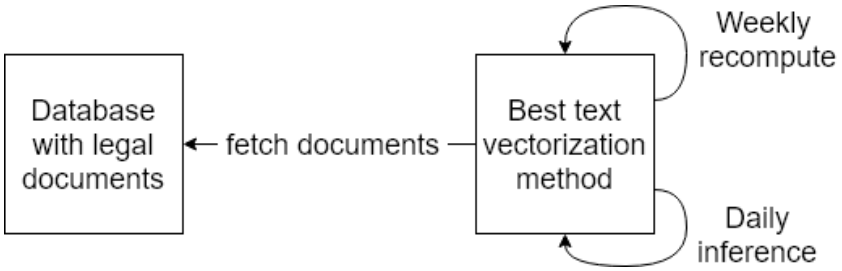


Figure 2.5: Second part of our proposed production system.

Based on the document database, many usable applications can be developed. Initially it would be valuable to develop a website that facilitates access to all of the scraped legal documents, including a traditional keyword search engine. Such a search engine can be extended to include search items based on our prototype. The third and last part of our production system implements this extended search engine by the workflow illustrated on in Figure 2.6. Like our prototype, the user selects a document from which they wish to find related documents. The selected document is given as input to the search engine, which utilizes the optimal text vectorization method and distance metric. The search engine finds the precomputed text vector for the input verdict and compute the neighboring verdicts in the vector spaces. The neighboring documents are thereafter returned as the search results to the user, ordered acceding in respect to their distance to the input document.

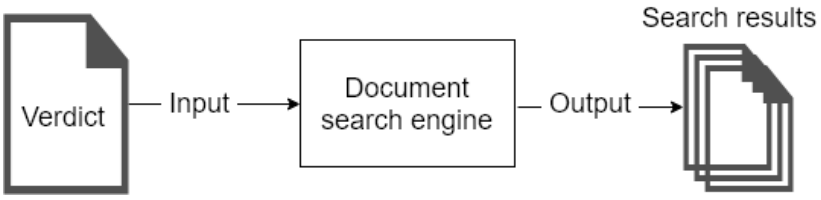


Figure 2.6: Third part of our proposed production system.

2.3 Data collection

We collected all publicly available tax verdicts from the websites afgoerelsesdatabasen.dk and skat.dk over a period spanning from the 29^{th} of February 2000 to the 31^{st} of December 2017. This web scraping resulted in the dataset summarized in Table 2.1.

Table 2.1: Summary of scraped verdicts.

Website	**Data formats**			**Total**
	Plain Text	**PDF**	**Empty verdicts**	
Afgørelsesdatabase.dk	15.326	777	16	16.119
skat.dk	12.182	0	10	12.192
Sum	**27.508**	777	26	28.311

Because we did not invest time in extracting the plain text from the 777 PDFs, and because the 26 empty verdicts were useless for our later document processing, we decided to discard these verdicts. As a result, we collected a usable data set of the 27.508 plain text verdicts scraped from the two websites.

It must be mentioned that our data set contained duplicate verdicts, both between the two websites, but also the individual websites contained multiple copies of the same document. However, based on our data analysis we estimate that out of 27.508 usable verdicts only about 100 are duplicates. We decided not to investigate the reason for these duplicates as it was outside the scope of the project.

2.4 Performance evaluation

The goal of this project was to produce a set of text vectors which captured the similarity between legal documents. To evaluate the text vectors produced by TFIDF and doc2vec, we first visualize the high dimensional vector space of the text vectors to look for groupings of the verdicts. Then, we perform clustering with silhouette score to evaluate how well

the verdicts group together in the high dimensional vector space, under the assumption that thematically similar documents would tend to group near one another given a good representation.

To evaluate the performance of a AI system which utilizes the text vectors, we contracted a human expert to annotate a small set of tax verdicts with relevance scores, later explained in Section 2.5.1. We calculated various statistical measures to evaluate to which extent our text representation methods captured similarity in the same vein as the human domain expert.

After concluding which text vectorization method best captured the annotated relevance, we utilize this method to generate a new set of documents for further annotation by the domain expert, as described in Section 2.5.2. We performed this new annotation to be able to conclude whether our new AI system provided better assistance than the current search results provided by the individual verdict websites.

2.5 Annotated data sets

Both of our annotated verdict data sets are scored by our Senior Tax Manager Søren H. Eriksen[5] from SEGES, later referred to as tax expert. As an expert in legal matters concerning taxation, Mr. Eriksen used his domain knowledge to annotate the verdicts with their relevance to one another, as a way to assess the extent to which the document vector embeddings aligned with the needs of potential users. The detailed process of constructing and annotating each of the two data sets is described in the following sections.

2.5.1 First relevance data set

The tax expert constructed the first annotated verdict data set by executing the following steps:

1. Selected a set of four tax problems he has worked with and has expert knowledge of.
2. Described each tax problem with a set of descriptive keywords.
3. Used these keywords to search the two verdict websites for relevant verdicts, using the keywords in an OR pattern (as the websites do not support the AND operator).
4. Documented the link and order of the first 10-15 search results from each of the two websites.
5. Read and annotated each verdict found in the searches with a relevance score, where 0 is irrelevant and 1 is relevant. Note: by choice, the tax expert assigned a score of 0.5 to a small subset of the documents to indicate partial similarity, but in our analysis we assigned scores of 0.5 to 0. This process resulted in a data set of 5 sets of keywords (i.e. origin verdicts) and a total of 101 candidate verdicts.

The resulting annotated data set contained 103 verdicts due to time constraints. An additional complication presented itself in the tax expert selecting certain verdicts for annotation which were outside of the collected document corpus, i.e. verdicts published after the 31^{st} of December 2017. Thus, some of the annotated verdicts had to be excluded from our performance evaluation. This exclusion resulted in 93 remaining verdicts, where 39 of the verdicts were annotated as relevant and 54 were annotated as irrelevant.

Because our AI system takes a whole text document as input for the similarity search, we transformed the annotated data using the following procedure: Each relevant search result are treated as the input (called an origin verdict) and the remaining verdicts of the search result - both relevant and irrelevant - are treated as the output (called candidate verdicts).

[5] sre@seges.dk - https://www.seges.dk/medarbejdere/lcsre

The search order from the original annotation data set is kept. This transformation process is illustrated via the example of initial data set seen in Table 2.2 which is transformed to the final data set seen in Table 2.3.

Table 2.2: Example of initial annotation data set.

Website	**Keywords**	**Link**	**Search order**	**Relevance**
Website1	Keyword1, keyword2	link1	1	0
Website1	Keyword1, keyword2	link2	2	0
Website1	Keyword1, keyword2	link3	3	1
Website1	Keyword1, keyword2	link4	4	1

Table 2.3: Same example after the transformation.

Origin link	**Candidate link**	**Search order**	**Relevance**
link3	link1	1	0
link3	link2	2	0
link3	link4	4	1
link4	link1	1	0
link4	link2	2	0
link4	link3	3	1

This transformation had the advantage of constructing a data set useful for comparing our text vectorization methods, and also increasing the size of the data set. The performed transformation resulted in a final annotation data set of 784 unique combinations of origins and candidates, where 398 and 386 of the candidates are relevant or irrelevant, respectively.

This uniform distribution of candidates that are relevant and irrelevant is unrealistic, as it is often the case for search systems that there are more irrelevant than relevant candidates. The uniform distribution in our data set is a result of the initial keyword search performed by tax expert, which have been performed using well chosen keywords based on an expert knowledge regarding the origin cases and problem domain.

2.5.2 Second relevance data set

To evaluate the ability of our chosen text representation method to align with user search goals, we asked the tax expert to select four origin verdicts (i.e. one document annotated as relevant from each of the four keyword searches performed when annotating the first data set), and then we generated 15 candidate verdicts for each of them to emulate the results of a search process using our AI system. Thus, we generated a total of 60 candidate verdicts, where 3 of them were already annotated in the first relevance data set. Therefore, the tax expert had to annotate the remaining 57 candidate verdicts with their relevance with respect to their origin verdict.

3 Results

In this chapter we present the results of our project. Our results are based on the computed TFIDF vectors and PVs using the hyper-parameter settings described in Section 2.1.

3.1 Text vector visualizations

Large datasets are confusing at the best of times. However, high-dimensional data has the intrinsic disadvantage that it is impossible to visualize, limiting human intuition. Notions of similarity and clustering - also referred to as the structure of the data - are hard to grasp. We turn to techniques developed in the field of dimensionality reduction to effectively represent high-dimensional text vectors in a two-dimensional space. Specifically, we employ the newly (i.e 2008) developed method of t-distributed stochastic neighbor embedding (t-SNE) to better grasp the structure of our text vectors[3].

t-SNE is able to model similar data points together and non-similar data points apart in a transformed lower dimensional space. Additionally, t-SNE is able to preserve local structures of the data while revealing global structures such as clusters. The intuition behind t-SNE is to represent the high-dimensional data via two (or three, etc.) dimensional surrogate points that preserve distances and local geometry between points in high dimensions. As such, the technique is close in spirit to embedding algorithms such as Multi-dimensional Scaling and ISOMAP, particularly the latter which uses preserves local geometry by embedding a neighborhood graph. Specifically, t-SNE creates the low-dimensional map by maximizing the probability that two neighboring high-dimensional points are neighbors in the low-dimensional space.

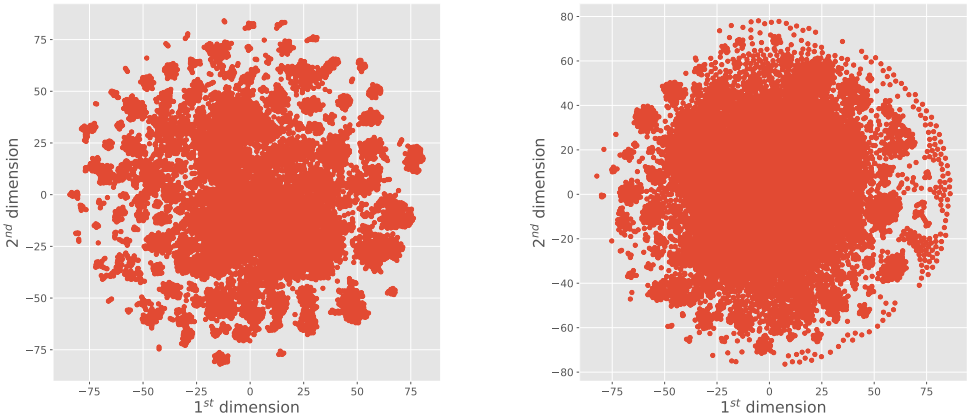
We utilized the t-SNE implementation[1] in scikit-learn version 0.19.1 with its default settings, except specifying the reduction to 2-dimensional vectors and the random state to 0 for reproducibility.

First we performed dimensionality reduction of our computed TFIDF vectors. However, due to their huge size of 339.188 dimensions it was infeasible to compute the t-SNE model on their full dimensionality. Therefore, we initially truncated the dimensions down to 200, using another dimensionality reduction method called singular value decomposition (SVD). SVD is used because it is more computationally efficient than t-SNE, but SVD has the disadvantage that it captures less of the local structures in the high-dimensional space. We utilized the SVD implementation[2] in scikit-learn version 0.19.1 with its default settings, except specifying the reduction to 200-dimensional vectors and the random state to 0.

After the SVD truncation we perform t-SNE on the 200 dimensional TFIDF vectors, and the resulting 2-dimensional vectors are plotted in Figure 3.1a. We also perform t-SNE on our 200 dimensional PVs and the resulting 2-dimensional vectors are plotted in Figure 3.1b.

Figure 3.1 shows that both our TFIDF and PVs vectors group well together into individual groups. However, we observe that our TFIDF vectors are more split into individual groups, but we assume this is due to the SVD truncation, and not representative of the high-dimensional TFIDF vectors.

[1] http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
[2] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html



(a) Visualization of the SVD and thereafter t-SNE dimensional reduced TFIDF vectors.

(b) Visualization of the t-SNE reduced PVs.

Figure 3.1: Visualization of the text vectors reduced to 2-dimensional vectors.

3.2 Cluster visualizations

To evaluate the how well the verdicts group together in the high dimensional vector space, we performed K-means clustering of their text vectors. However, the optimal number of clusters is unknown prior to computation of the K-means algorithm, as the user-defined parameter K restricts the K-means algorithm to always find K clusters. Therefore, to obtain the optimal clustering, we compute K-means multiple times over a range of K values. For each clustering we compute the resulting silhouette score, which is a measure of how well a clustering satisfies the two properties of cohesion (i.e. a point's distance to the mean of its cluster) and separation (i.e. a point's distance to the means of all the other clusters). The silhouette score ranges from -1 to $+1$, where a high value indicates that the clustering satisfies both cohesion and separation.

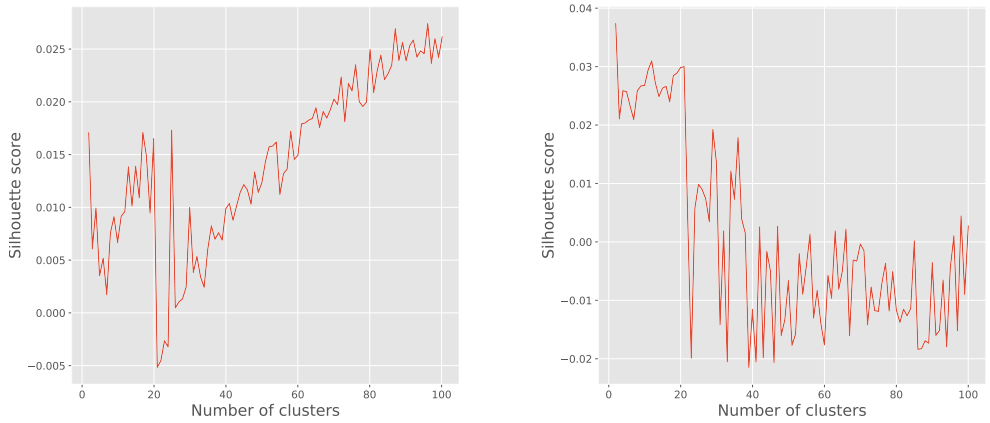
We perform K-mean on the high dimensional vectors, i.e. the 339.188 dimensional TFIDF vectors and and 200 dimensional PVs with an increasing K from $K = 2$ to $K = 100$. The resulting silhouette score per clustering is plotted in Figure 3.2. We used the K-means[3] and silhouette score[4] implementations in scikit-learn version 0.19.1 with their default settings, except a random state set to 0 for both functions.

Figure 3.2a shows that the silhouette score of our TFIDF vectors increases as the number of clusters increases. The score flattens out as the number of clusters moves to 100; however, the optimal number of clusters is inconclusive, especially due to the low maximum score of 0.027. We assume the inconclusive results are due to the high dimensionality the TFIDF vectors in combination with a clustering - where $K < 100$ - that is not well separated or cohesive. In contrast, the silhouette score of the PVs, plotted in Figure 3.2b shows the expected knee in the score where the optimal clustering is found. However, the maximum score is still low, as it is approximately 0.037.

To visualize the clusterings with the maximum silhouette score, we present the two clusterings of each text vectorization method with the highest score, as seen in Figure 3.3 and 3.4 for the TFIDF vectors and in Figure 3.5 and 3.6 for the PVs.

These visualized clusterings show that the trained t-SNE vectors capture the global structure of both the high dimensional TFIDF vectors and PVs, as the groups of verdicts

[3] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
[4] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html



(a) Silhouette score per clustering of TFIDF vectors.

(b) Silhouette score per clustering of PVs.

Figure 3.2: Visualization of the text vectors reduced to 2-dimensional vectors.

close to another tend to be members of the same cluster. The only plot deviating is Figure 3.5 of the PVs clusterings with the highest score.

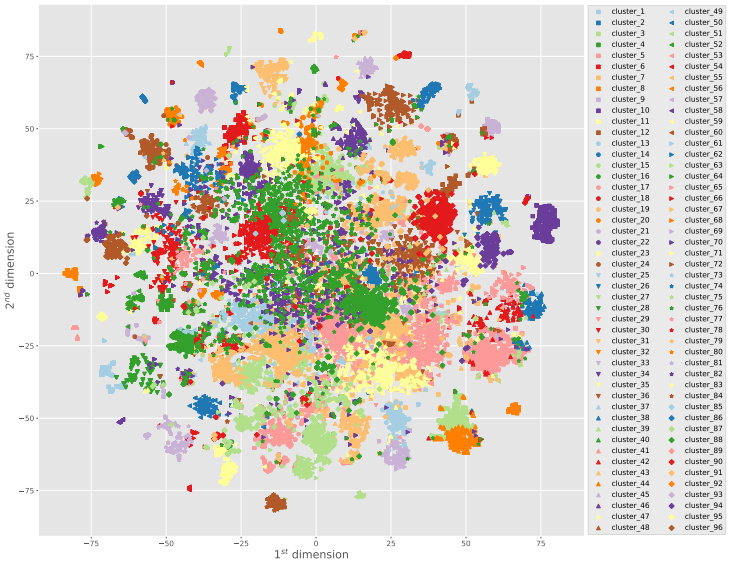


Figure 3.3: Visualization of the t-SNE reduced TFIDF vectors with cluster labels of 96 clusters with the highest silhouette score of 0.0274

3.3 Verdict relevance evaluation

For our analysis of how well our text vectors captured the relevance between verdicts, we used our first annotated data set, described in Section 2.5.1.

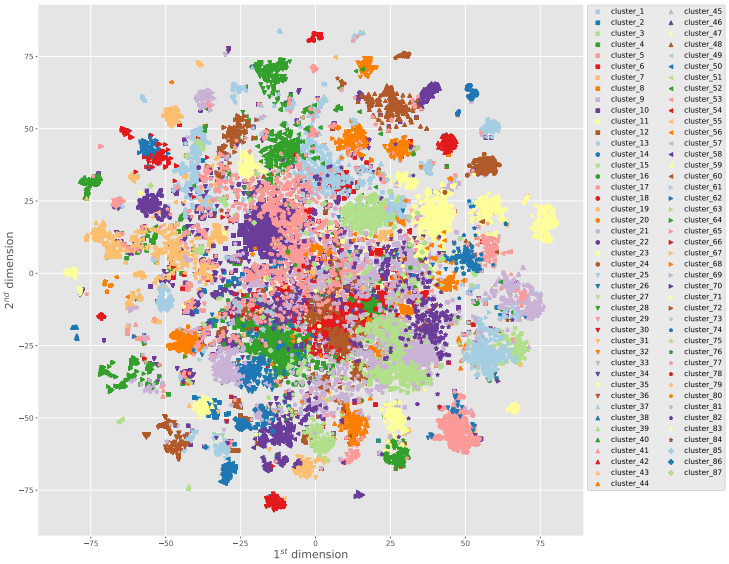


Figure 3.4: Visualization of the t-SNE reduced TFIDF vectors with cluster labels of 87 clusters with the second highest silhouette score of 0.0269

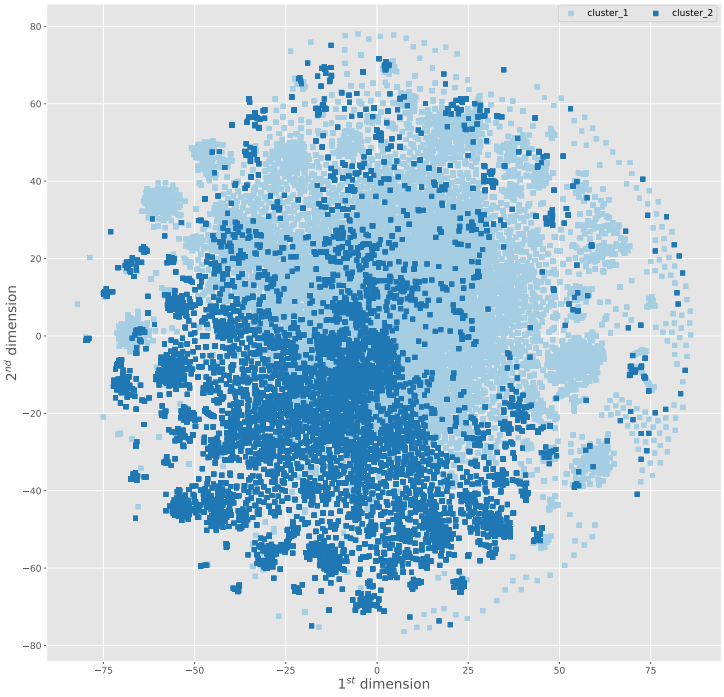


Figure 3.5: Visualization of the t-SNE reduced PVs with cluster labels of 2 clusters with the second highest silhouette score of 0.0373

3.3.1 Distance analysis

We computed the distances from each origin verdict to each of its candidate verdicts for all four combinations of our utilized text vectorization methods and distance metrics. The violin plots shown in Figure 3.7 present an overview of the differences in distances from origins to respectively irrelevant and relevant candidates.

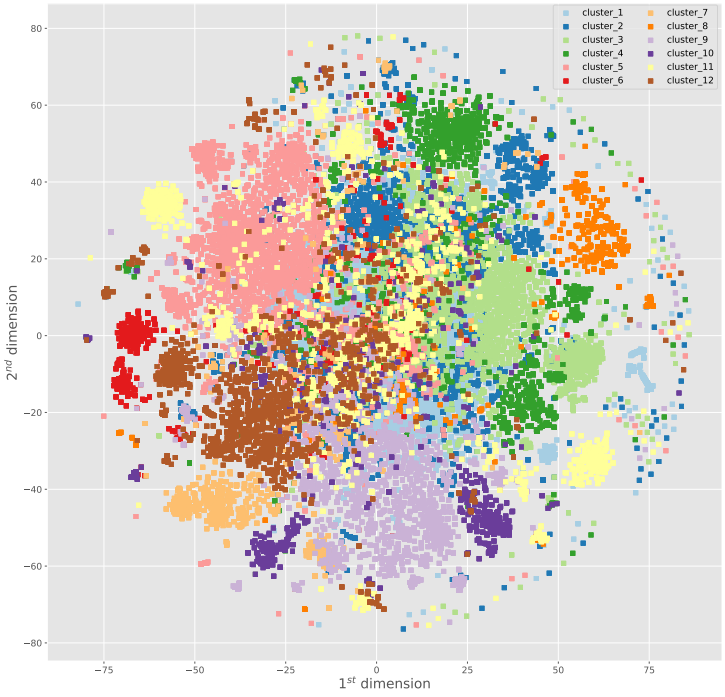


Figure 3.6: Visualization of the t-SNE reduced PVs vectors with cluster labels of 12 clusters with the second highest silhouette score of 0.0309

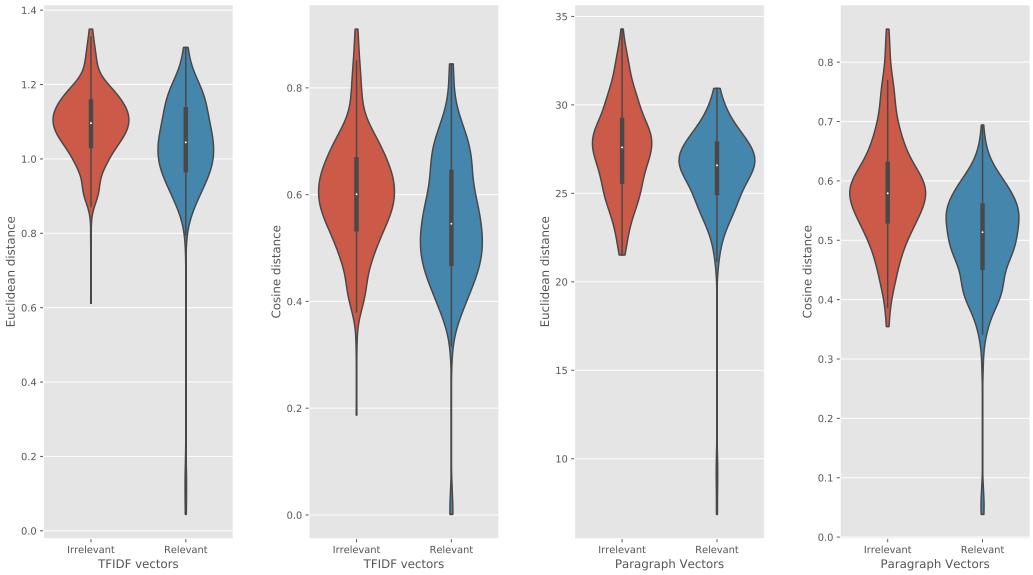


Figure 3.7: Violin plots of distances from origins to irrelevant and relevant candidates.

3.3.2 Nearest neighbor analysis

Using the computed distances between all 27.508 verdicts in the scraped data set for all four vectorization and distance metric combinations, we can rank the verdicts based on their distance to each origin, in the annotated data set. This ranking represents the order of the nearest neighbor (NN) search results our AI system would return if each origin was

given as input. Based on this NN search ranking, we present the violin plots shown in Figure 3.8 for the four system combinations. These plots show the difference between the irrelevant and relevant candidates with respect to where they are in the order of the search result per origin.

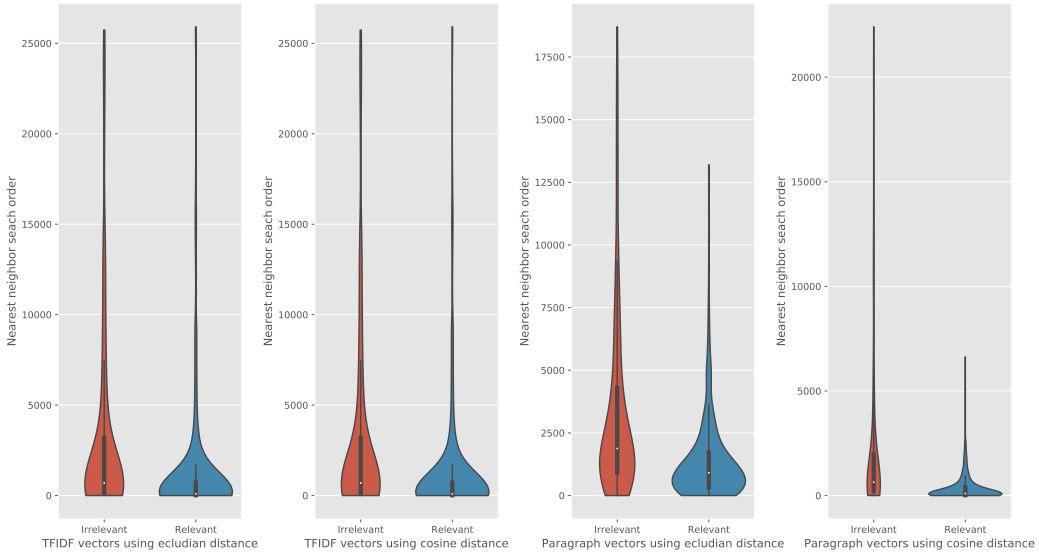


Figure 3.8: Violin plots of NN search order of origins and irrelevant and relevant candidates.

All plots show the positive result that the text vectorization methods capture document similarity, both in terms of the distance analysis and NN search analysis, in the same vein as the human domain expert. This is evident by the visible tendency that the relevant candidates have shorter distances to their origin compared to the irrelevant candidates.

As seen in Figure 3.8 the system using our PVs and the cosine distance metric returns the best search results with respect to our annotated data set, since both relevant and irrelevant candidates have favorable search rankings. We conclude such a system will be the best choice for implementing in a production setting.

3.3.3 Statistical questions

We present four simple statistical questions and answer them as a method to explain the quality of the search results returned by the best system, i.e. the system using our PVs and the cosine distance metric. The questions as a follow:

- **Is the NN of each origin relevant or irrelevant?** The distribution of relevant and irrelevant candidates as the NN, in the annotation data set, is 81% and 19% respectively.
- **How big should the result set be before at least one relevant candidate will appear for 75% of the searches?** The result set only needs to contain 13 verdicts before 75% of the searches return at least one of the annotated relevant candidate.
- **How big should the return result set be before half of the relevant candidates will appear for 75% of the searches?** The result set needs to contain 560 verdicts before 75% of the searches will include at least half of the relevant candidates.
- **How big should the result set be before all of the relevant candidates will appear for 75% of the searches?** The result set needs to contain 1625 verdicts before 75% of the searches will return at all of the annotated relevant candidate.

3.4 Search method comparison

For our analysis of comparing the keyword search utilized by the tax expert to our new NN search, we use both annotated datasets (see sections 2.5.1, 2.5.2). However, from the first data set we only used the initial 103 verdicts found by the tax expert.

To compare the two data sets, we had to merge the search results from the two different websites and restrict each set to only contain the first 15 verdicts. Thus the final annotated data set of keyword searches contained four searches each with 15 verdicts as search results, like the the second annotated data set containing the NN search results.

We compare the two search methods on how many relevant verdicts on average are found the in search result. The keyword search returned in average 7.75 relevant verdicts per search, where as the NN search returned in average 1.75 relevant verdicts per search.

Additionally, we present the distribution of the found irrelevant and relevant verdicts over the search result order, as illustrated in the violin plot in Figure 3.9.

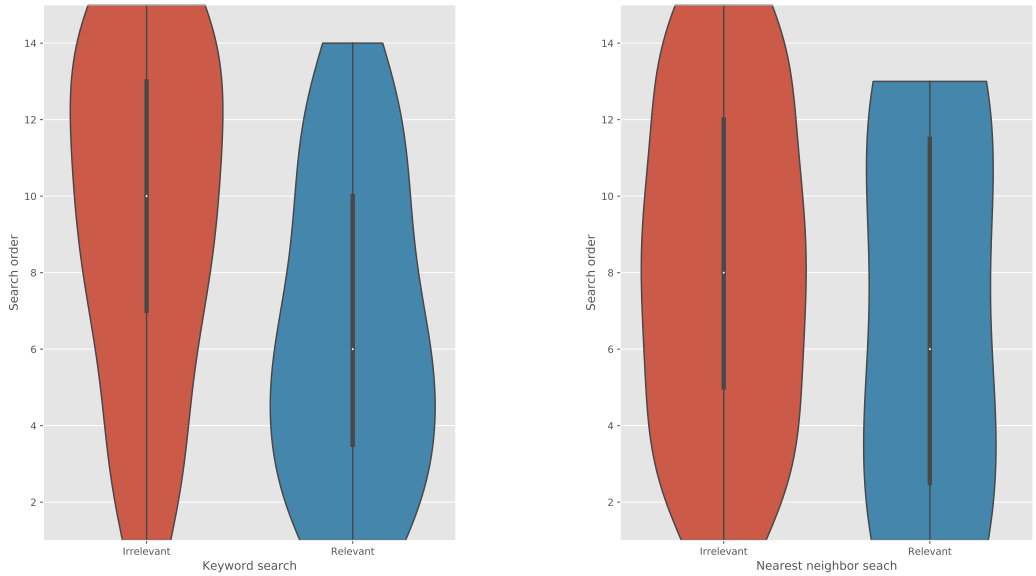


Figure 3.9: Violin plots illustrating the order of the irrelevant and relevant verdicts found in the search result for both the keyword and NN search method.

4 Discussion

In this chapter we conclude our project with the following discussion of the results presented in Chapter 3. Additionally, this chapter contains a summarization of potential future work, as seen in Section 4.1.

Our developed prototype of a nearest neighbor (NN) document search engine shows positive results. Throughout Chapter 3 the results show that both of our chosen text vectorization methods captured groupings of legal verdicts, and that this grouping for some of our annotated verdicts represents the relevance relationship between them. However, it is inconclusive if these findings can be generalized to the whole corpus, mainly due to the unsupervised nature of our text vectorization methods, and also the limited size of our annotated data sets. Thus, the grouping of our learned text vectors can also express other relationships than the relevance relationships we seek in this project. Future data mining or data annotation has to be performed before finding such additional patterns in the collected data set.

Based on our collected data set and learned text vectors, we conclude the NN document search engine should be constructed by the use of the paragraph vector (PV) method and the cosine distance metric. In particular, the nearest neighbor analysis shows the benefit of using the PVs and cosine distance as it returns the relevant verdicts much earlier in the search result compared to the other method combinations.

Our comparison of the current utilized keyword search and our new NN document search, presented in Section 3.4, shows that our method is usable as a supplement to the current method. We conclude that our search method does not return as many relevant verdicts as the current search method. However, the two methods are not exactly comparable, given that keyword search requires the user to define a specific set of search parameters while our NN search only requires the user to provide a existing verdict. Therefore, we assume the good results for the current search method is mainly because of our annotated data set was made by a tax expert who knows which keywords are suitable for getting the relevant verdicts. To investigate this assumption, we need a larger annotated data set from multiple users with different levels of experience.

As seen in Section 2.2, we also proposed a production system valuable to the legal advisers. This system contains an end-to-end solution that: (1) scrapes the documents from the world wide web, (2) learns the text vectors, and (3) presents the search engine functionality to the end user through a website. Our developed prototype mainly concerns step 2 of training the text vectors. However, it is critical to understand that the value for the end user is mainly created in step 1 where the documents on the world wide web are collected in a single database and step 2 where the users can search to gain an overview of the whole corpus. This new system is faster for the user as the current process, which involves visiting multiple websites (and Google) to search for the relevant documents. One can view our NN document search as a supplement to the traditional keyword search, and as such we advise that such a keyword search functionality should be the first to be developed.

To summarize our findings in this project we answer the following problem statements, presented in Chapter 1:

- **How can the tax verdicts, publicly available on the world wide web, be given as input to an artificial intelligence (AI) system?** Using text vectorization methods one can represent multiple documents as vectors in a vector space, forming the basis of NN based document search.

- **How can an AI system predict the relevance of previous verdict to a new legal case?** We have shown that using NN search of text vectors an AI system can predict the wanted relevance between document. Additionally, our results show that the quality of these predicted relevances are depended on ability of the text vectorization methods to capture the relationship between documents in the vector space.
- **How accurate are such predictions?** Using our small annotated data set, we presented an accuracy in terms of returned relevant verdicts give the size of the returned verdict set. Our best search method (i.e. PV and cosine distance) will only need to return a total of 13 verdict for it to include at least one relevant candidate in 75% of the searches.
- **Can such an AI system provide better assistance than the current search results provided by the individual verdict websites?** Based on our annotated data sets we conclude our NN document search functionality is outperformed by a domain expert employing keyword search. However, our results also show that our search functionality can find relevant verdicts, and therefore it can be useful for inexperienced users in the legal domain. However, most potential users will be experienced in the domain which will diminish the actual benefit of our search functionality.

4.1 Future work

We present the following future work in the prioritized order reflecting our view on their importance, difficulty and value for the end user:

1. Develop a production system similar to the one described in Section 2.2.2. The main focus should be on the data engineering task of scraping all legal documents from the world wide web on a periodically basis, and merging all of the data sets into a single database. Based on this database a traditional keyword search can be facilitated to the users.
2. Utilize data from other sources on the world wide web, e.g. the official legal guidance published by SKAT[1], the tax related messages published by SEGES[2], or the actual law text[3].
3. Laws can change over time, making previous verdicts irrelevant even thought they concern the same problem as as the current case question. We do not handle the problem in this project, but it is essential for the end user that a solution is developed. Including the current law description as documents in to the relevance search could provide a fast way for the legal advisor to get the overview of the current law related to the case.
4. Improve the evaluation of our NN document search by annotate a larger data set containing other legal documents from multiple domains and annotated by multiple users.
5. Modify the PV method such that the training of the PVs incorporates user-specified relevance judgments as a supervision signal.
6. The collected tax verdict data sets could be used for many new interesting projects, to name a few:

[1] http://skat.dk/skat.aspx?oid=124&ik_navn=footer
[2] https://www.landbrugsinfo.dk/oekonomi/skat/skattemeddelelser/sider/startside.aspx
[3] https://www.retsinformation.dk/

a) Analyze the additional attributes of the scraped data set using the trained text vectors, e.g. answering questions such as: do the clustering of the text vectors match the grouping of the verdicts based on the data attributes `source` or `judicial_authority`.

b) Train new machine learning models to solve problems such as generating a text summary of the long verdicts, or automatically annotate each verdict with the winning part of the lawsuit.

List of Acronyms

AI	artificial intelligence	1, 5, 7, 9, 12, 13, 19, 23, 24
FNN	feedforward neural network	8
NLP	natural language processing	7
NN	nearest neighbor	1, 9, 19–21, 23, 24, 28
PV	paragraph vector	7–9, 15–20, 23, 24, 28
SVD	singular value decomposition	15, 16
TFIDF	term frequency–inverse document frequency	7–9, 11, 15–18, 28
t-SNE	t-distributed stochastic neighbor embedding	15–19, 28

List of Figures

List of Tables

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[3] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.